**Batch: C3 Roll. No.: 16010123217**

**Experiment: 8**

**Grade: AA / AB / BB / BC / CC / CD /DD**

|  |
| --- |
| **Title:**  Study of Graph traversal methods DFS and BFS |

**Objective:** To understand graph as data structure and methods of traversing Graph

**Expected Outcome of Experiment:**

|  |  |
| --- | --- |
| **CO** | **Outcome** |
| **CO2** | Apply linear and non-linear data structure in application development |

**Websites/books referred:**

1. <https://ds1-iiith.vlabs.ac.in/exp/breadth-first-search/index.html>
2. <https://ds1-iiith.vlabs.ac.in/exp/depth-first-search/index.html>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Abstract**:

A graph is a data structure composed of a set of nodes (called vertices) connected by edges. It can be used to represent various relationships and structures in the real world, such as networks (social, communication, etc.), navigation paths, and more. Each edge connects two vertices, illustrating the relationship between them. Graphs can be directed (edges have a direction) or undirected (edges do not have a direction), and can be weighted or unweighted.

**Algorithm**

**DFS**Depth first search (DFS) algorithm starts with the initial node of the graph G, and then goes deeper and deeper until we find the goal node or the node which has no children. The algorithm, then backtracks from the dead end towards the most recent node that is yet to be completely explored. The data structure which is being used in DFS is stack.

* **STEP 1**: Start by putting any one of the graph's vertices on top of a stack (acts as source node of DFS).
* **STEP 2**: Take the top item of the stack and set its visited as 1.
* **STEP 3**: Create a list of that vertex's adjacent nodes. Add the ones whose visited is 0 to the top of stack.
* **STEP 4**: Keep repeating steps 2 and 3 until the stack is empty.

**BFS**

The algorithm starts with examining the source node and all of its neighbours. In the next step, the neighbours of the nearest node of the source node are explored. The algorithm then explores all neighbours of all the nodes and ensures that each node is visited exactly once and no node is visited twice.

**STEP 1:** Set visited as 0 for all nodes in the Graph.

**STEP 2:** Enqueue the selected source node into the queue.

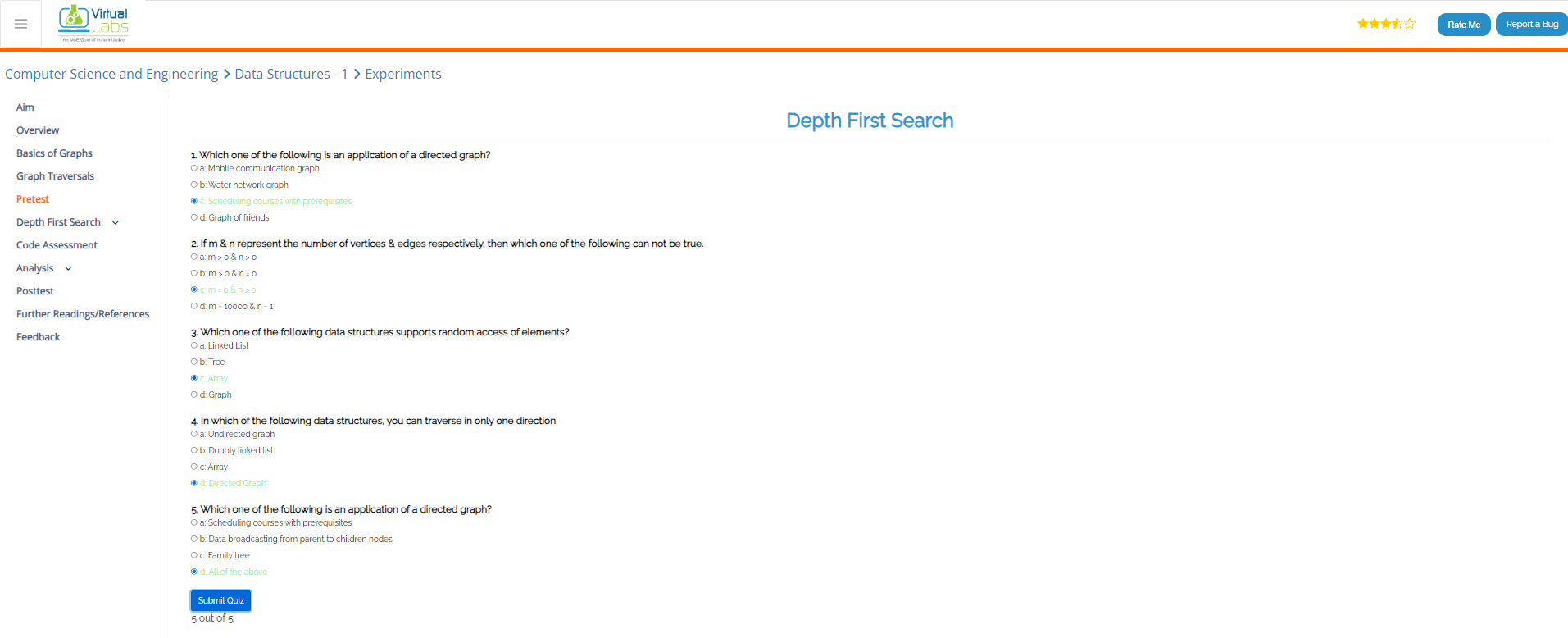
**STEP 3:** Dequeue a node N from queue and update its visited as 1.

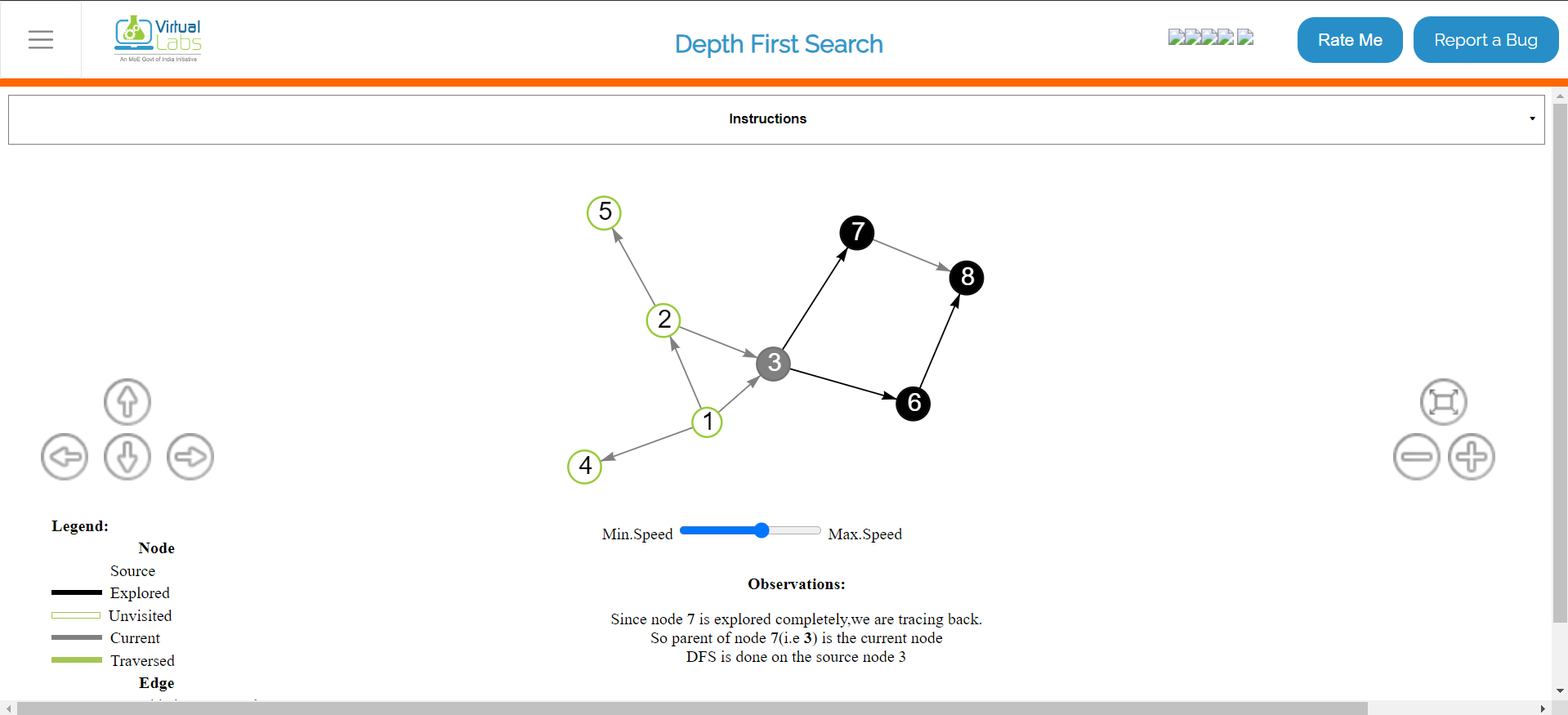
**STEP 4:** Enqueue all the neighbours of node N which are not present in the queue and whose visited is 0.

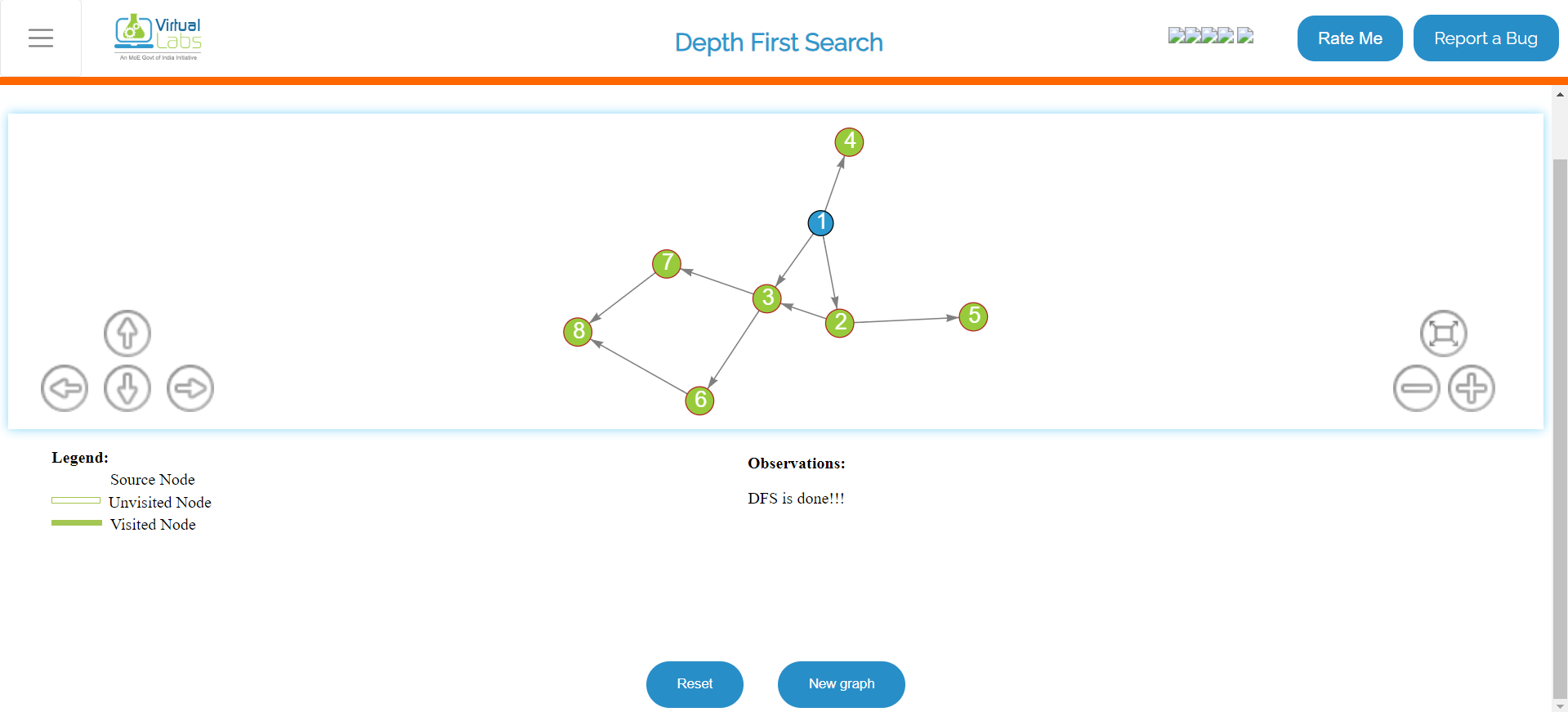
**STEP 5:** Repeat steps 3 and 4 until queue is empty.

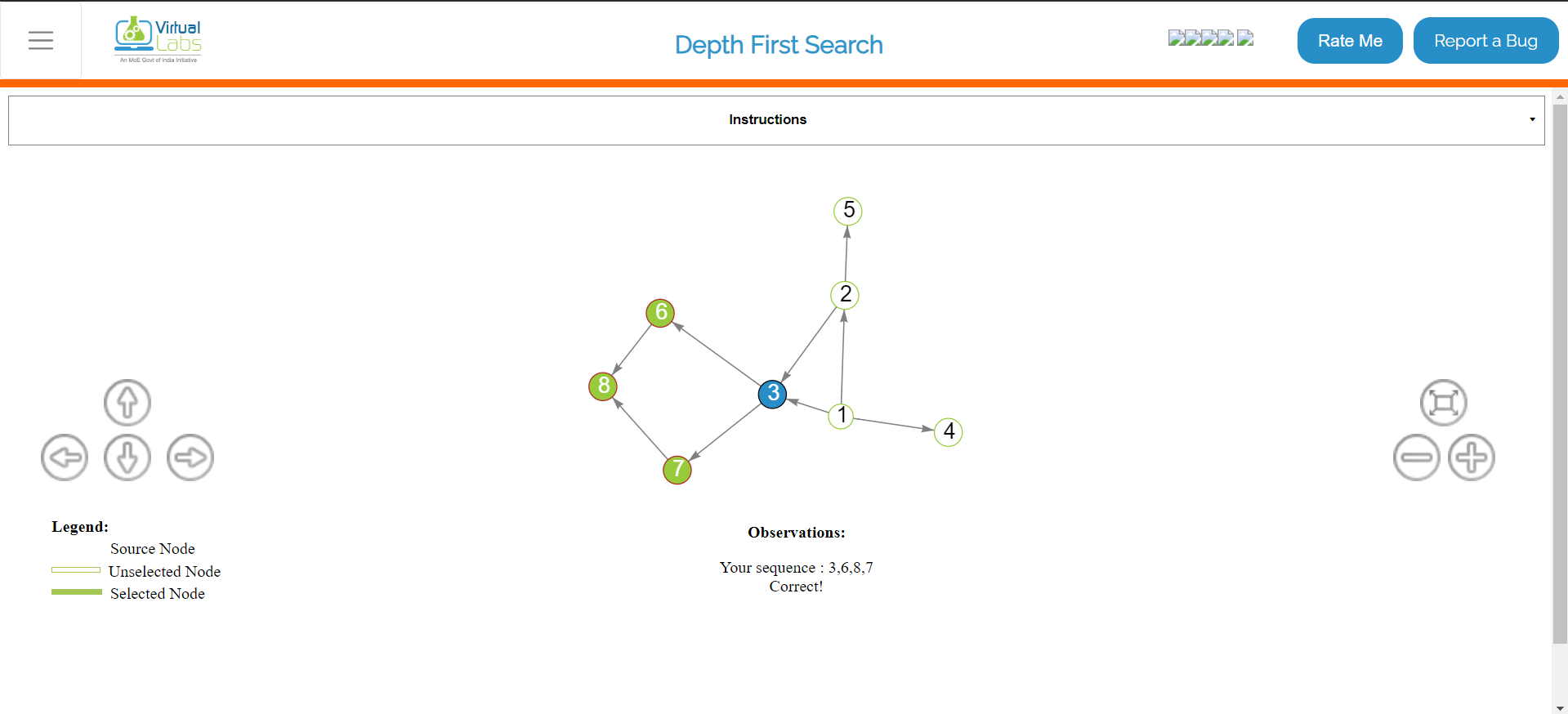
**STEP 6:** EXIT

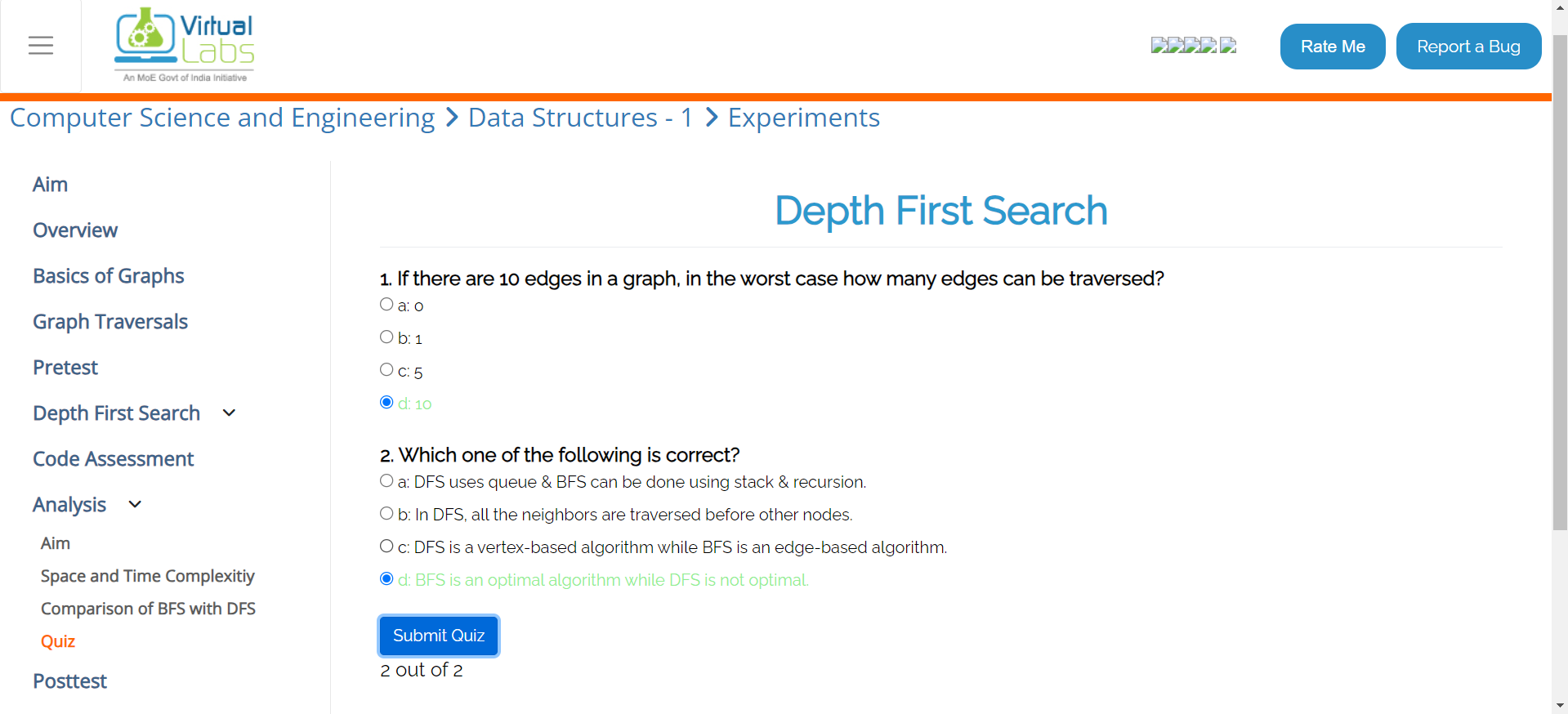
**Virtual Labs**

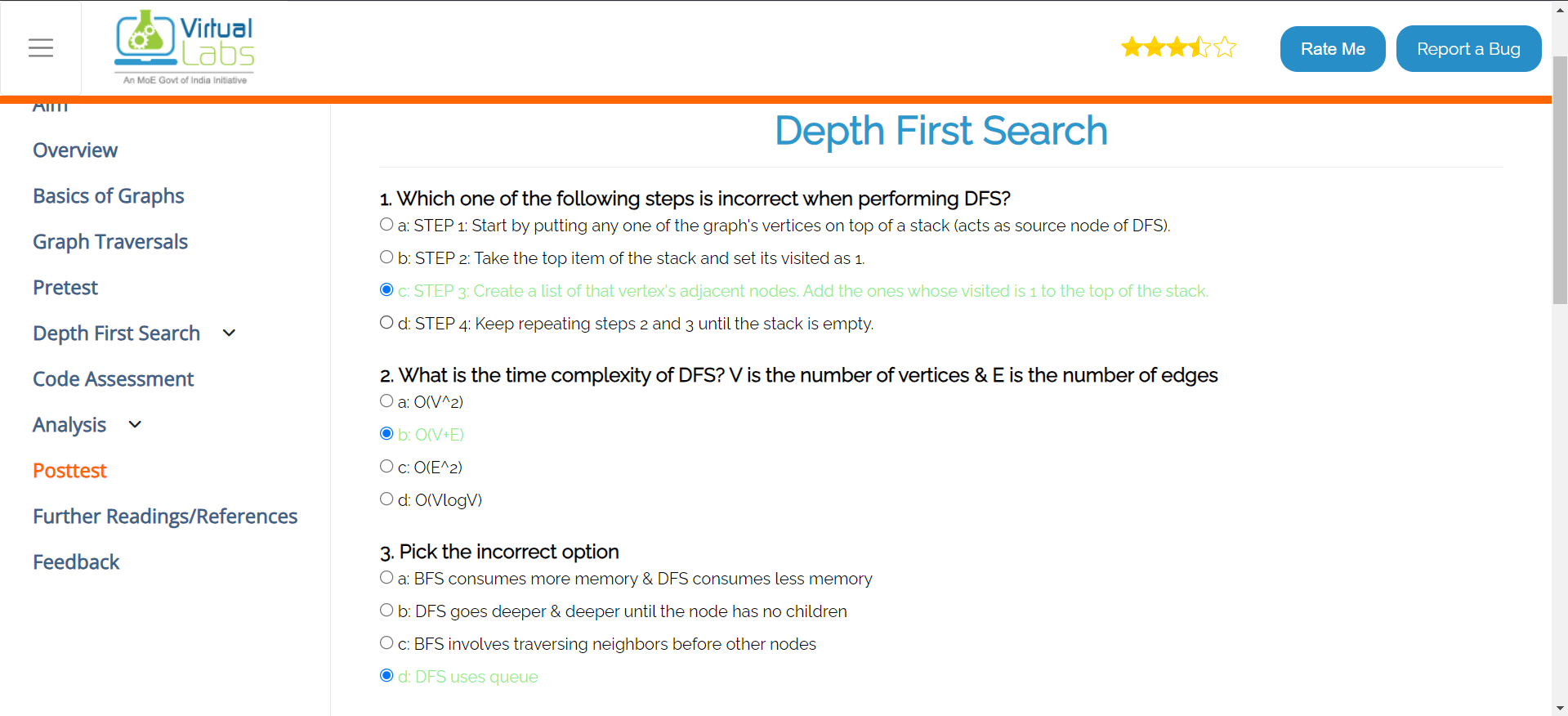
**DFS  
**

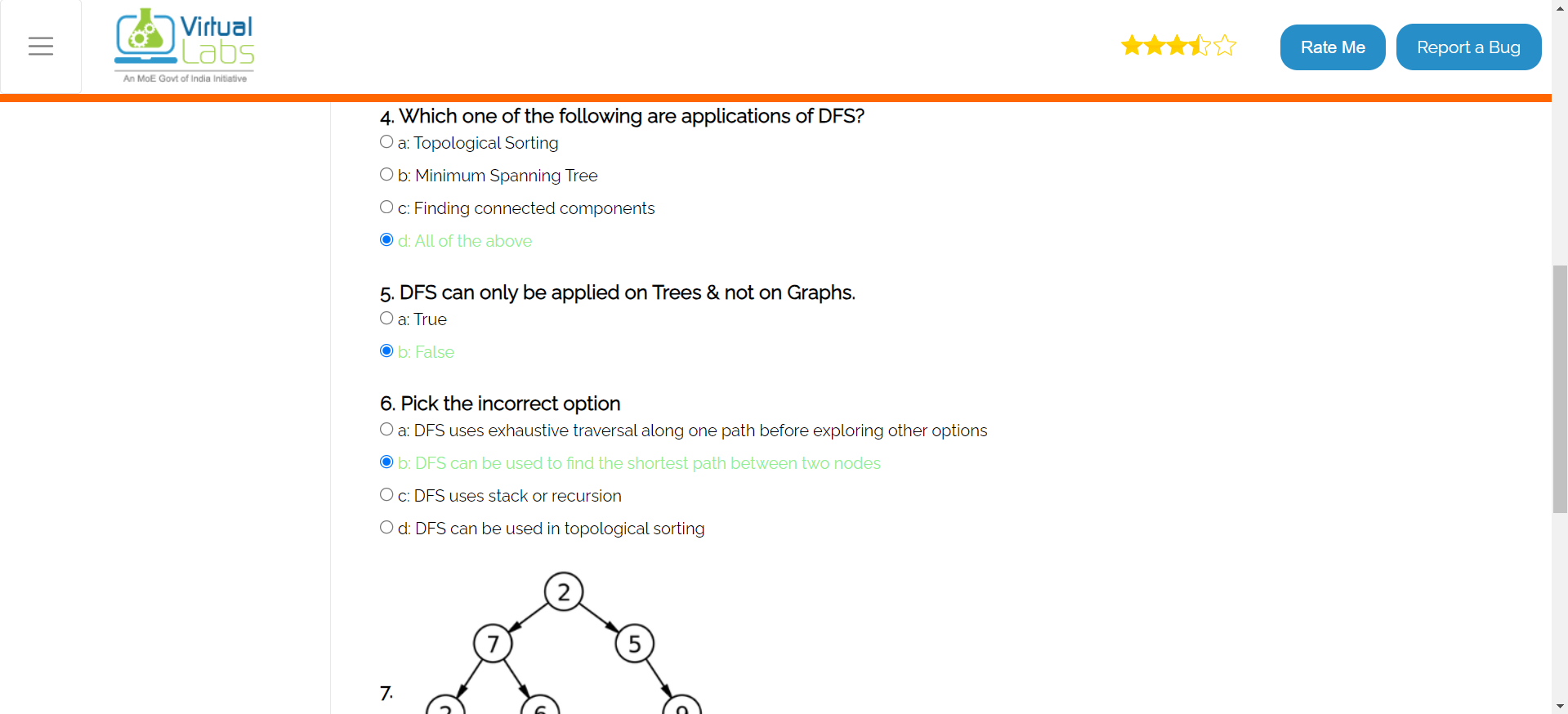
****

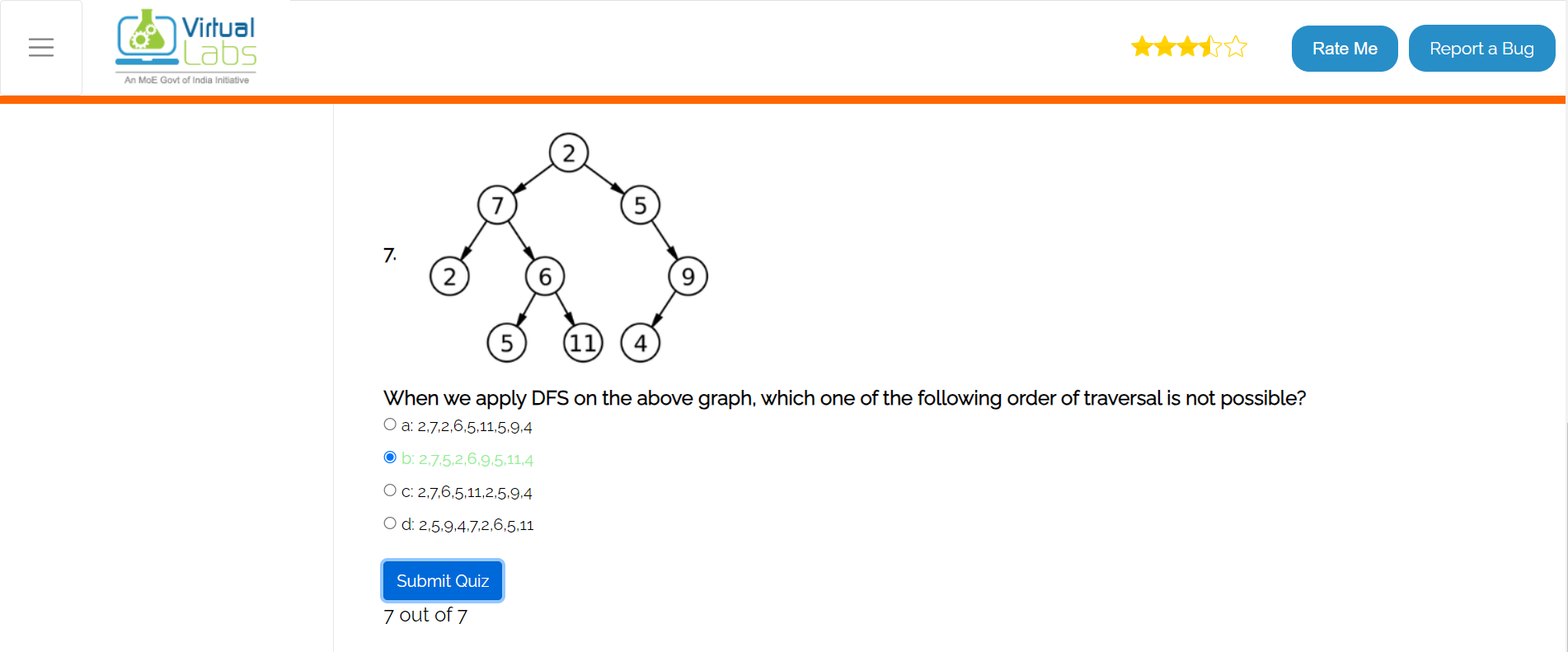
****

****

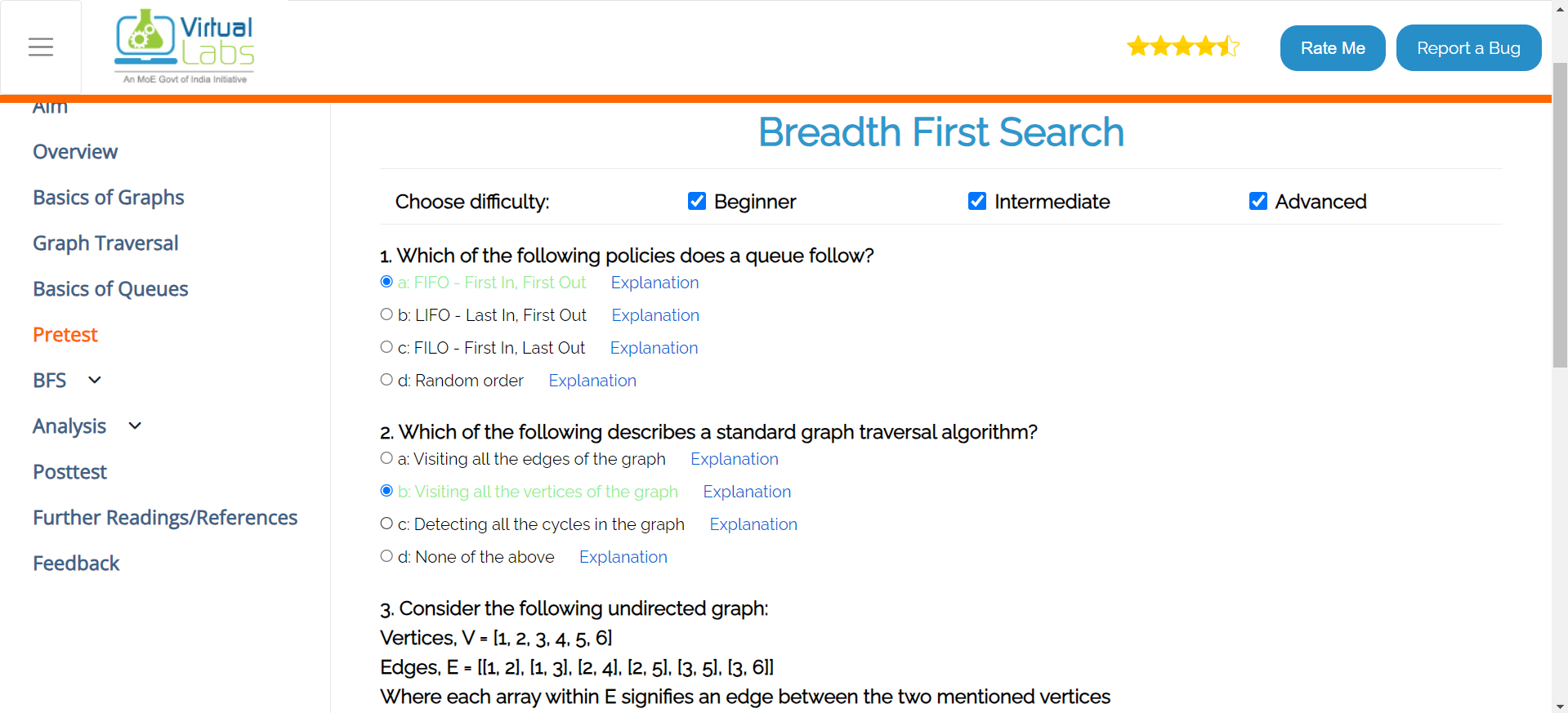
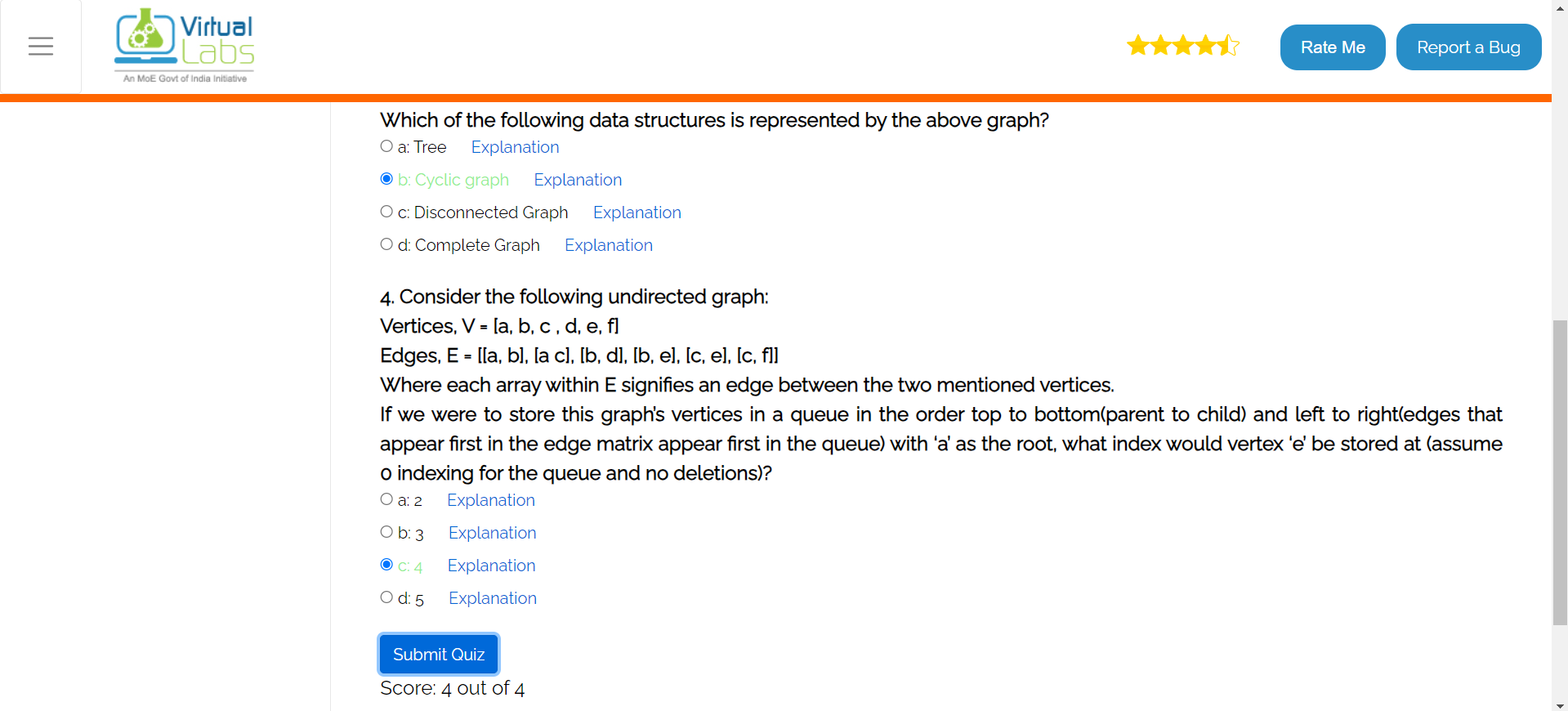
****

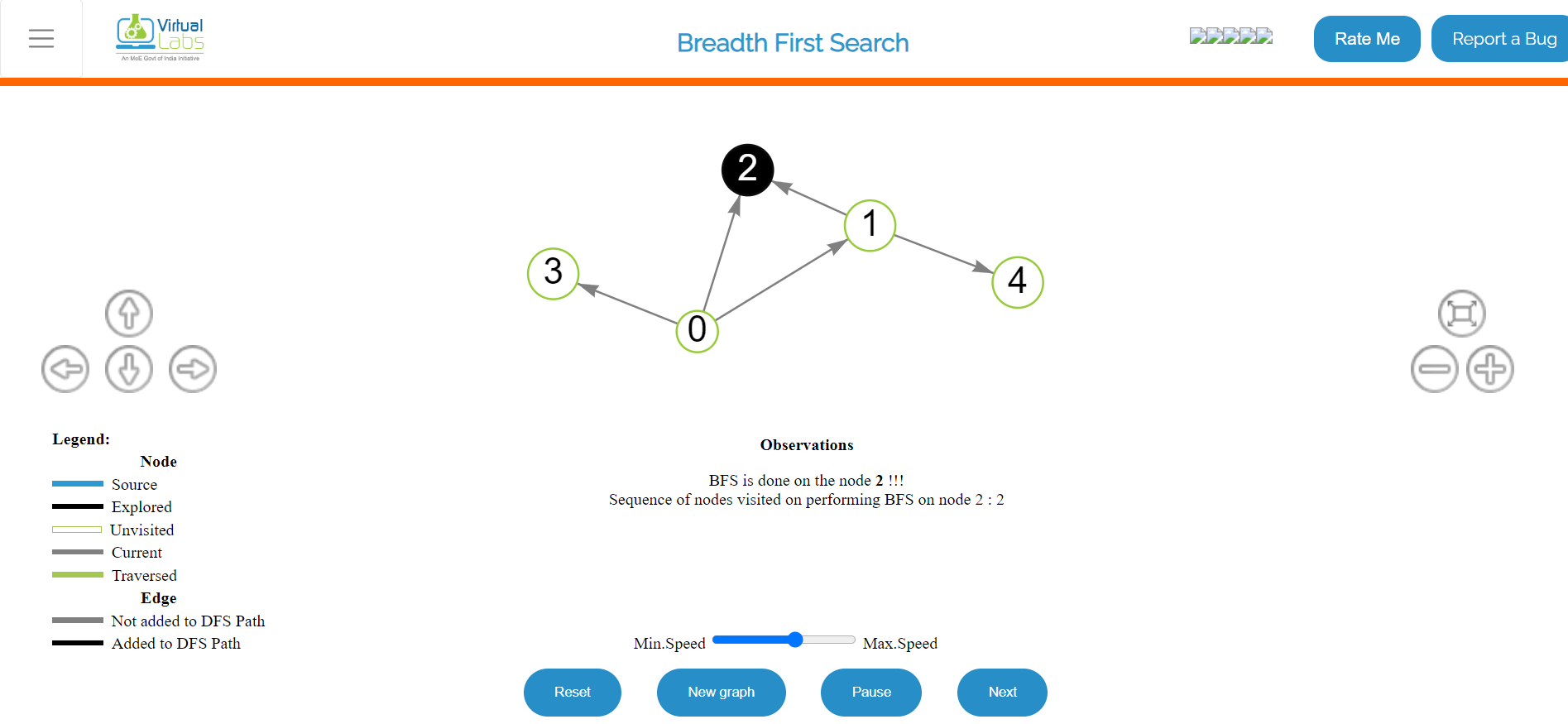


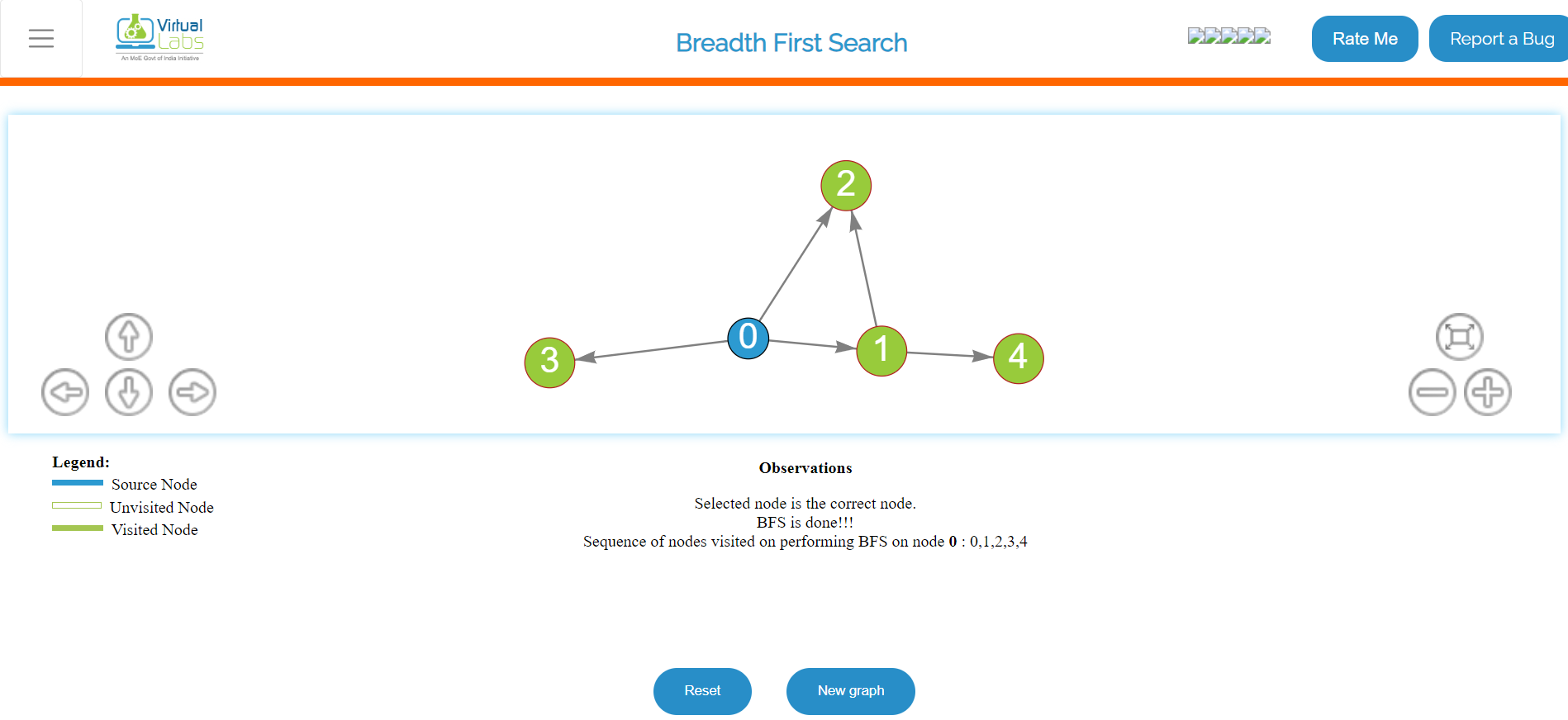


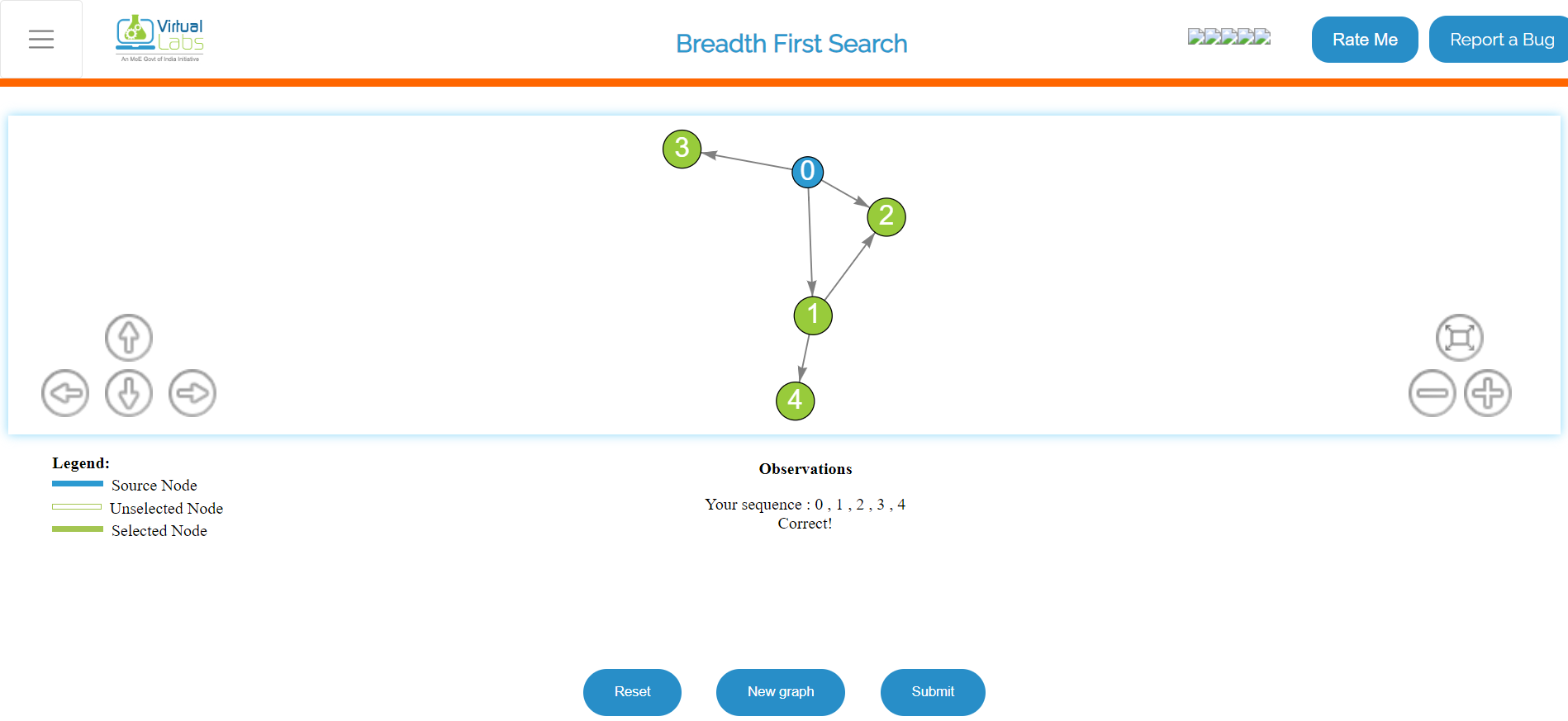


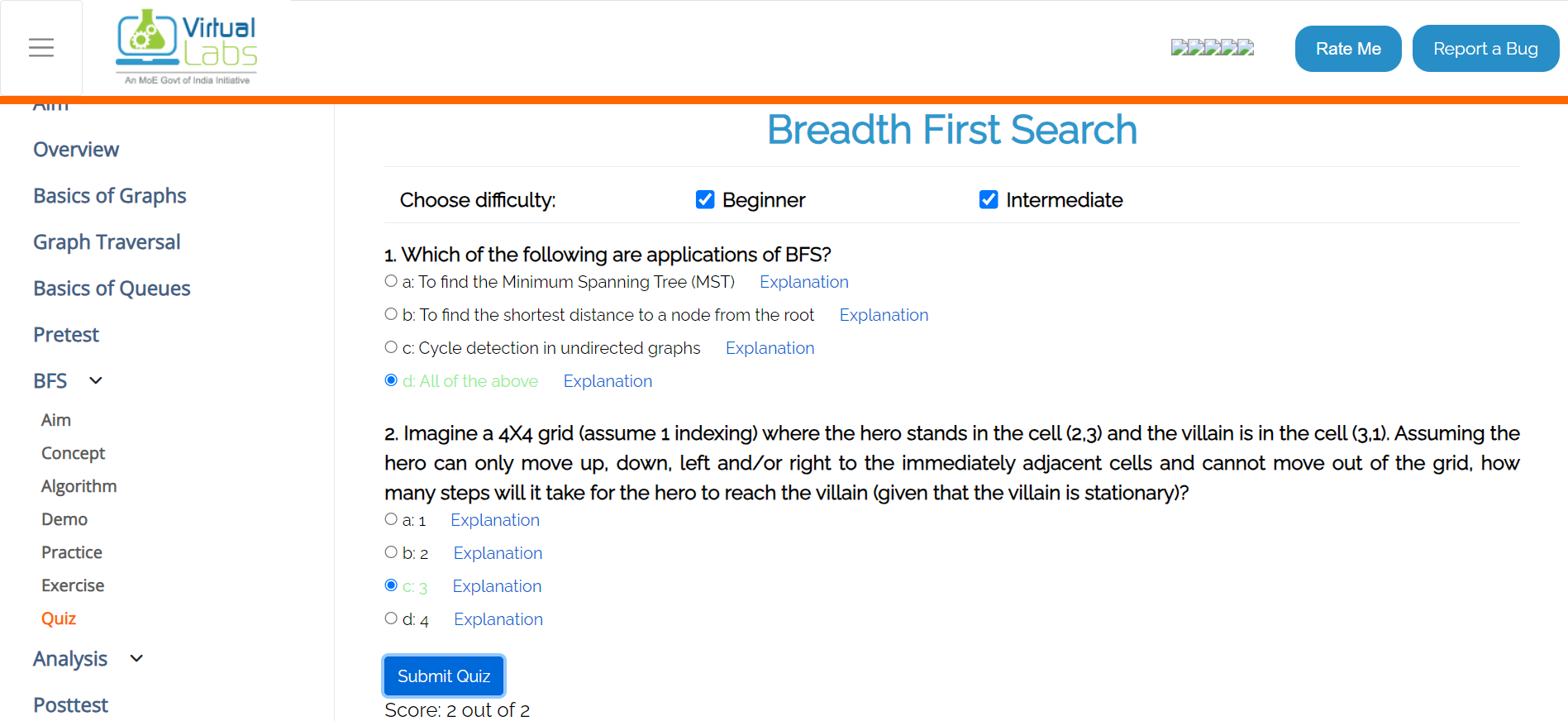
**BFS**

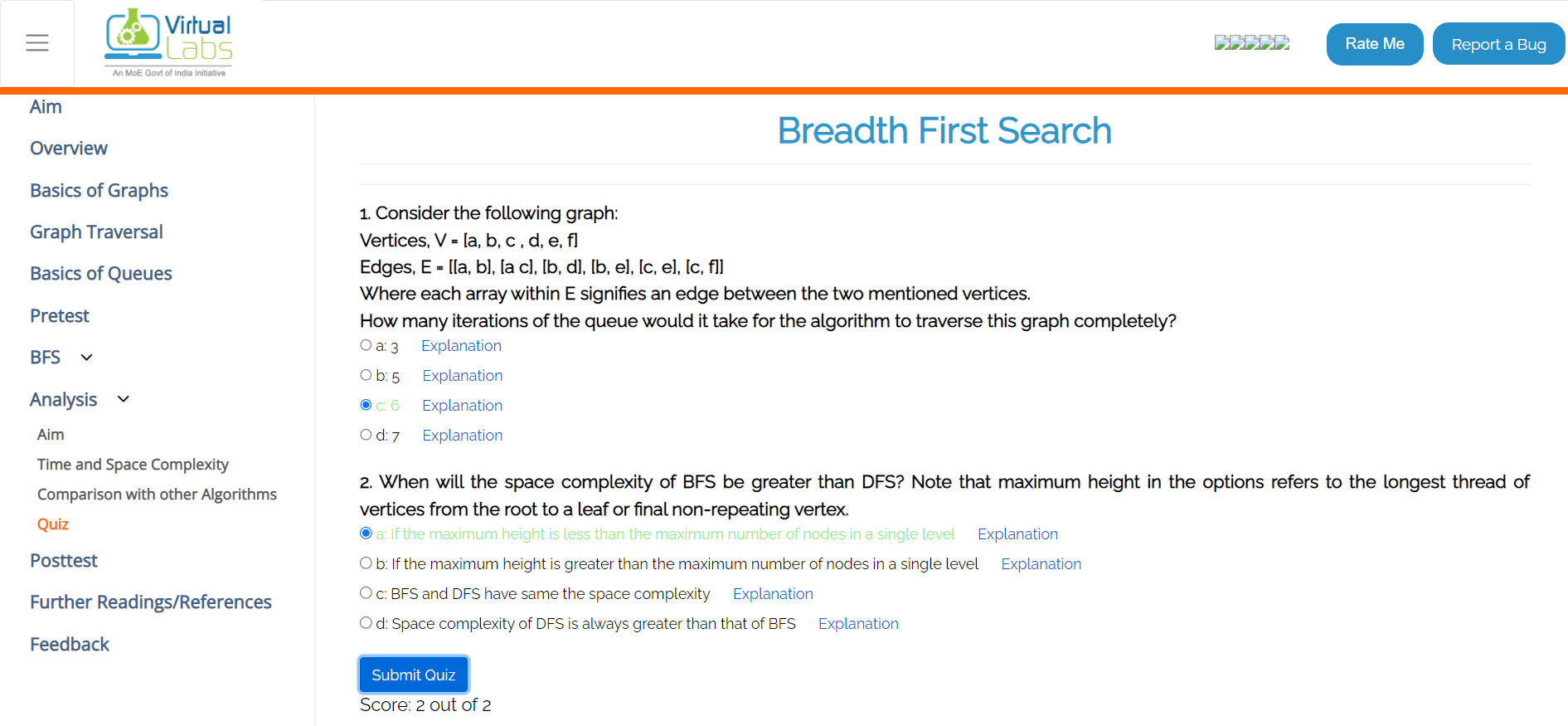
****

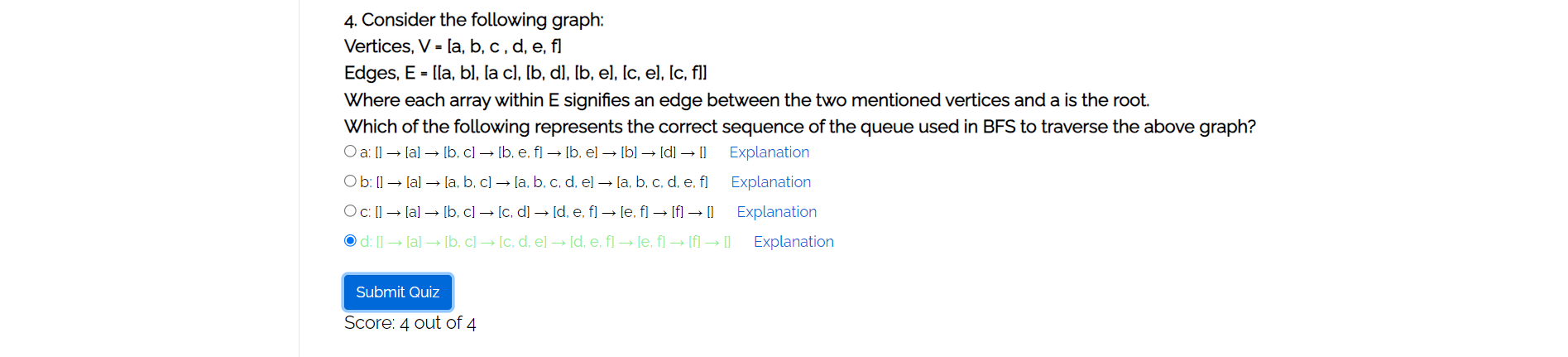
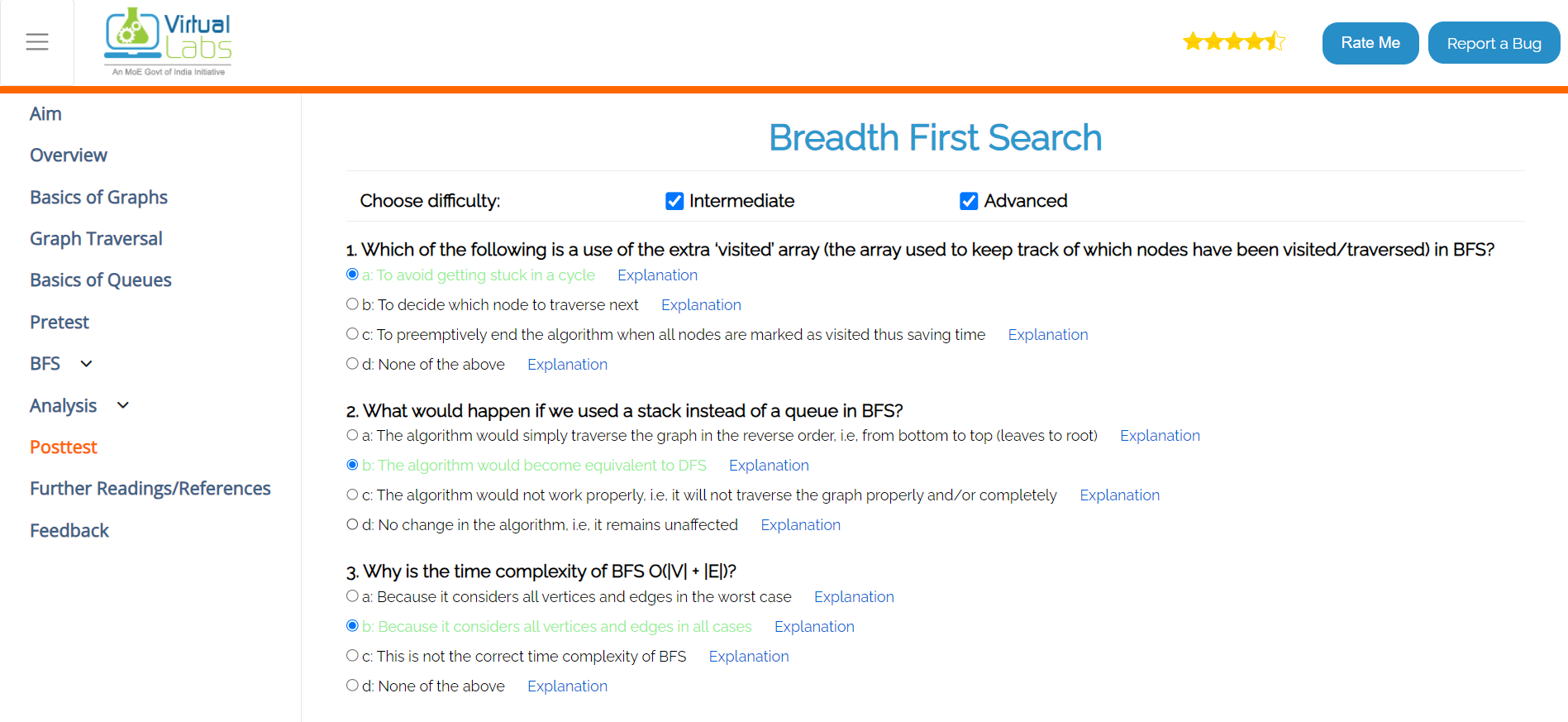
****

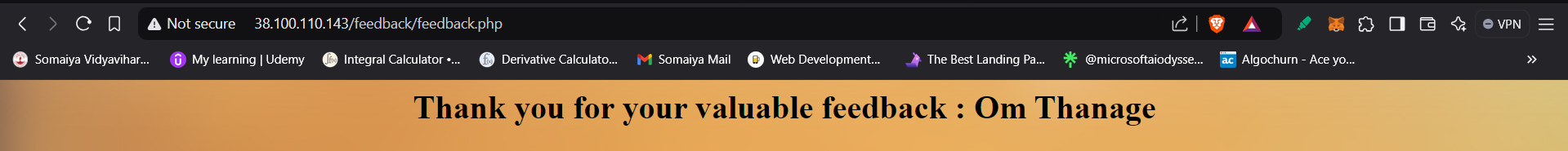
****

****

****

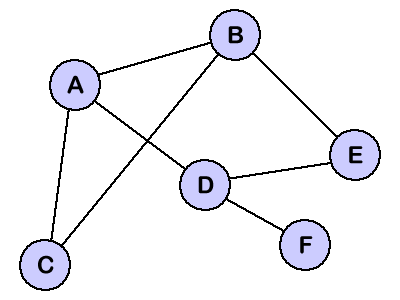
****



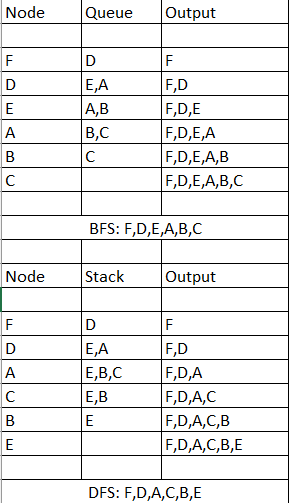
****

**Solution of Assigned numerical problems using BFS and DFS:**

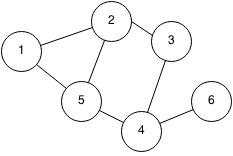
**Q1)**

****

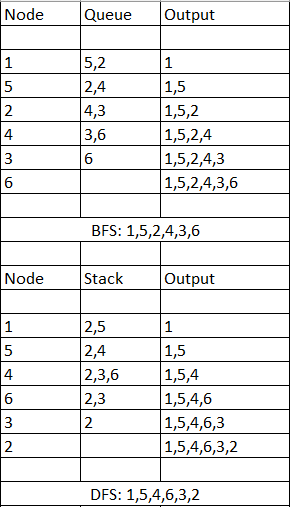
**Start : F**

****

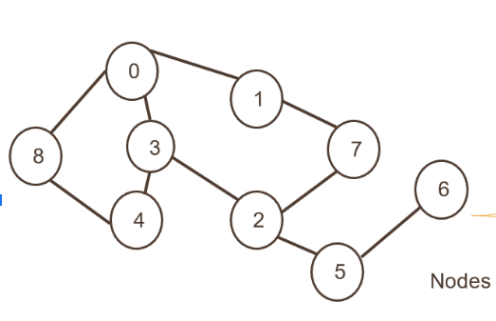
**Q2)**



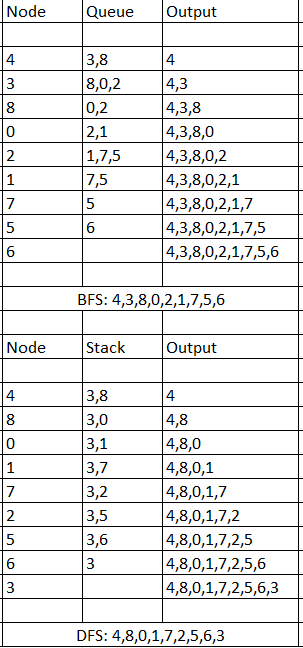
**Start : 1**



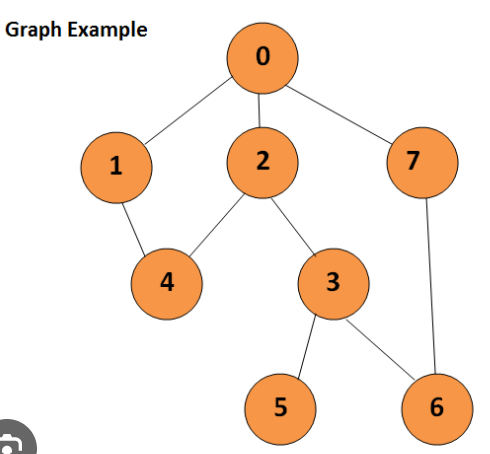
**Q3)**



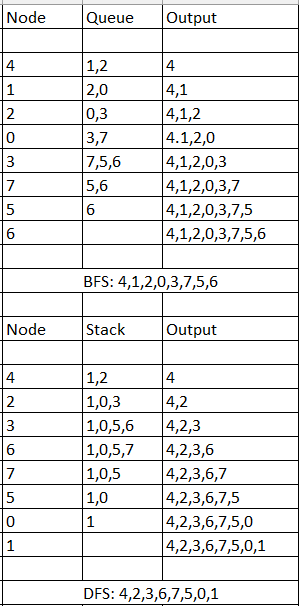
**Start : 4**

****

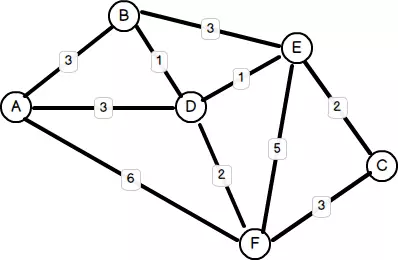
**Q4)**



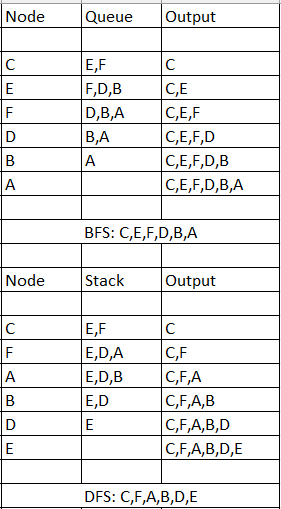
**Start : 4**

****

**Q5)**



**Start : C**

****

**Conclusion: -**

The experiment on DFS and BFS using vlab showed that BFS is great for finding shortest paths in unweighted graphs, while DFS is better for exploring depths and detecting cycles. Vlab made it easy to visualize and understand their differences and applications.